Journal of Korea Multimedia Society Vol. 20, No. 3, March 2017(pp. 511-519)
https://doi.org/10.9717/kmms.2017.20.3.511

PLC and Arduino Interaction Based on Modbus Protocol

Yunju Jeong[†], Md Israfil Ansari[††], WooHyeon Shin[†††], Bonggu Kang[††††], JinSeop Lim[†††††], HyeonSik Moon[††††††], Jaechang Shim[†††††††]

ABSTRACT

This Paper introduces the design and communication method between PLC (Programmable Logic Controller) and Arduino based on MODBUS Protocol. MODBUS connection can be established in a new or existing system very easily, therefore we used this protocol in our proposed system. In the field of automatic devices, multi-function serial port such as RS232, RS422, RS485, and so on creates a great convenience to the developer. This proposed system used RS485 as a key mediator for data exchanging on a connected network. We also believe that it will reduce the development cost in various automated industry because this system can be reused or can be implemented any such PLC installed machines. RS485 is used as a communication interface between PLC (as a slave) and Arduino (as a master), through which a reliable network is created for safe and fast communication. Furthermore, RS485 allows multiple devices(up to 32) to communicate at half duplex on a single pair of wires and provides a long connectivity area (up to 1200 meters) as compare to other device, which makes it a user-friendly for various devices in the automated industry. Moreover, Arduino can play as a mediator by connecting third party device and setup a communication network with PLC.

Key words: PLC, RS485, MODBUS Protocol, Arduino, IoT

1. INTRODUCTION

Automation in electronic, machinery and many such companies are being adopted in a numerous way[1, 2]. Cost reduction, increase in productivity are the primary benefits cited with the help of automation. When it comes to cost reduction, we believe for a system which can be reusable or adaptable in various devices. This proposed system focuses on PLC based machines through which a communication network is setup with Arduino with the help for MODBUS Protocol.

PLC is a digital controller which is widely used in automation industry for electromechanical processes. PLC is designed for multiple arrangements of digital and analog inputs and outputs which extend temperature ranges and immunity to electrical noise. PLC is mostly used to control machinery,

※ Corresponding Author : Shim Jaechang, Address: (36729) Gyeongdong-ro 1375, Andong, Gyeongsangbuk-do, Korea, TEL : +82-10-9770-5645, FAX : +82-54-820-6164, E-mail : jcshim@anu.ac.kr
Receipt date : Sep. 9, 2016, Revision date : Jan. 3, 2017
Approval date : Feb. 13, 2017
[†] School of Computer Science & Engineering, Kyungpook National University (E-mail : vrjung@hanmail.net)
[††] Department of Computer Engineering, Andong National University (E-mail : israfila3@hotmail.com)
[†††] Department of Computer Engineering, Andong National University (E-mail : withfox6051@naver.com)
[††††] Department of Computer Engineering, Andong National University (E-mail : dngustls@naver.com)
[†††††] Department of Computer Engineering, Andong National University (E-mail : jinsseof@naver.com)
[††††††] Department of Computer Engineering, Andong National University (E-mail : wnl668@naver.com)
[†††††††] Department of Computer Engineering, Andong National University
※ This work (Grants No.C0398612) was supported by Business for Cooperative R&D between Industry, Academy, and Research Institute funded Korea Small and Medium Business Administration in 2016.

PLC comes with build in communication port, usually 9-pin RS-232, RS-422, RS-485 and Ethernet. Here we will be discussing on RS-485 port. Today modern PLCs can be programmed in a variety of ways, from ladder logic to any programming languages. Under the IEC 61131-3[3 ,4] standard PLC can be programmed using standard based programming languages, in which ladder logic is most commonly used. Ladder logic sometimes called "relay logic" was designed to work like relay logic, an old programming language which was already in wide use. It is run by reading and writing from the memory table of inputs, outputs and intermediate values.

Arduino is an open-source platform which is used for building electronics projects. The main point of "Arduino Platform" is to allow for easy and fast Platform. It is famous for its simple programming and rich resources and can also be mount each other like bricks to add some more modules. Arduino hardware platform has the power and resets circuitry setup with circuitry to program and communicate with the microcontroller over USB. It consists of microcontrollers and IDE which helps to upload computer code to the physical board. It uses a simplified version of C++, making it easier to learn and use[5, 6]. It is a flexible micro-controller and development environment that can not only be used to control devices but also can be used to read data from all kind of sensors. This leads Arduino to the development of a variety of hardware extensions and software libraries that enable wired and wireless communication with the Internet[7].

2. MODBUS PROTOCOL OVERVIEW

Modbus protocol is a serial communication protocol where digital communication networks can be implemented over long distance effectively and also in electrically noisy environments. Moreover, Modbus can run over virtually in all communication media, including twisted pair wires, wireless, fiber optics, Ethernet, telephone modems, cell phones and microwave. Here multiple receivers can be connected in a linear network where the only one will be master device and rest will be a slave. Slave can't transmit data without a request from the master node, and also can't communicate with another slave. The master node issues a MODBUS request to the slave nodes in two modes i.e. unicast mode and broadcast mode[8].

At the time of communication on MODBUS network. The protocol determines how the controller knows about its device address, recognize message address to it, determine the action to be taken and extract the data or information contain in the message. MODBUS protocol takes place at level 2 of the OSI model. At the physical level, MODBUS uses different physical interface like RS485, RS232 etc. RS485 on a Two-Wire interface

Table 1. Comparison of Modbus ASCII and Modbus RTU

	MODBUS ASCII		MODBUS RTU	
Characters	ASCII 0…9 and A...F		Binary 0…255	
Error check	LRC(Longitudinal Redundancy Check)		CRC(Cyclic Redundancy Check)	
Frame start	Character ':'		3.5 chars silence	
Frame end	Characters CR/LF		3.5 chars silence	
Gaps in message	1 sec		1.5 times char length	
Start bit	1		1	
Data bits	7		8	
Parity	Even/odd	none	Even/odd	none
Stop bits	1	2	1	2

which is most commonly used. At Data link layer MODBUS Serial Line Protocol is defined[10]. MODBUS RTU network consists of one “master” and up to 247 “slaves”[11].

In Table 1, we have presented the difference between Modbus ASCII and Modbus RTU.

2.1 MODBUS Addressing

The MODBUS address space includes 256 different address.

The Address 0 is for broadcast purpose and all slave must recognize the broadcast address. The Master node has no specific address.

Table 2. MODBUS Address

0	1 - 247	248 - 255
Broadcast Address	Slave Address	Reserved

2.2 MODBUS Frame Structure

Modbus frame is composed of ADU(Application Data unit) which encloses a PDU(Protocol Data Unit). ADU composed of Address, PDU and Error check. PDU is composed Function code and Data.

3. MODBUS OVERVIEW

3.1 MODBUS Function Codes

MODBUS has 10 function codes as shown in Table 4. In this proposed system we have used two codes i.e. 04 and 16.

A. Function 01: Read Coil Status

Function 01 reads ON/OFF status of discrete outputs in a slave. In other words, it is used to Read Coil Status or digital status from a Modbus Slave data acquisition device. It read from 1 to 2000 contiguous status of coils in a remote device. The Request PDU specifies the starting address i.e. the address of the first coil specified. The Query message specifies the starting coil and quantity of coils to be read. Table 5 Represents a request to read slave (20-27) from device 17. The coil in the response message are generally packed as one coil per bit of data field, Table 6 shows the frame structure.

Table 4. MODBUS Function Codes

CODE	Description
01	Read coil status
02	Read input status
03	Read holding registers
04	Read input registers
05	Force single coil
06	Present single registers
07	Read exception status
15	Force multiple coils
16	Write holding/multiple registers
17	Report slave ID

Table 5. Query Structure of Function 01

Field Name	RTU
Header	None
Slave Address	11
Function	01
Starting Address Hi	00
Starting Address Lo	13
No. of Points Hi	00
No. of Points Lo	25
CRC	--

Table 3. MODBUS Frame Structure

START	ADDRESS	FUNCTION	DATA	CRC CHECK	END
T1-T2-T3-T4*	8 BITS	8 BITS	N*8 BITS	16 BITS	T1-T2-T3-T4*

For T1-T2-T3-T4, 3.5 character times at no communication.
The address field contains the slave address
The function code indicates server for kind of action to perform
CRC (cyclic redundancy checking) checks the content of the entire message.

Table 6. Response Frame Structure of Function 01

Field Name	RTU
Header	None
Slave Address	11
Function	01
Byte Count	05
Data (Coils 27-20)	CD
CRC	--

B. Function 03: Read Holding Registers

The Modbus RTU Function 03 is used to Read Holding Registers (4x register references), in a Modbus slave device. This function reads continuous blocks of holding registers in the remote device. Registers are addressed starting at Zero.

Query message specifies the starting registers. Here we present an example of a request to read register 40108-40110 from slave device 17. Query frame structure is shown in Table 7. In the Response message, the register data are packed as two bytes per registers with binary contents. For each register, the first byte contains the high order bits and the second contains the low order bits. Response Frame structure is represented in Table 8.

Table 7. Query Structure of Function 03

Field Name	RTU
Header	None
Slave Address	11
Function	03
Starting Address HI	00
Starting Address LO	6B
No, of Points HI	00
No. of Points LO	03
CRC	--

Table 8. Response Frame Structure of Function 03

Field Name	RTU
Header	None
Slave Address	11
Function	03
Byte Count	06
Data HI (Register 40108)	02
Data LO (Register 40108)	2B
CRC	--

C. Function 04: Read Input Registers

The Modbus RTU function 04 is used to read Input registers from a Modbus Slave device. Function 04 reads the contents of the 3X registers. Here the address starts from 0(zero), for example, 1-10 are addressed as 0-9. Function 04 includes the quantity of registers to read from the Modbus Slave device. With Function 04 master used to request one or more holding register value from the device but only one slave device can be addressed in a single query. The frame structure of query and response are shown in Table 9 and Table 10 respectively.

In Table 9, Top 2 byte in Data field is the start address register, the lower two bytes is word units. In Table 10, the first byte in Data field is the number of data to read (byte unit) and then the top data, sub-data and lower data.

D. Function16: Write Holding Registers

With Function16 Master Device write values into a sequence of holding registers to specified slave.

The query message specifies the register reference. It writes values which are specified in the

Table 9. Query Structure of Function 04

Field Name	RTU
Header	None
Slave Address	0A
Function	04
Data	00 00 00 02
Register Request	0001
CRC	B298

Table 10. Response Frame Structure of Function 04

Field Name	RTU
Header	None
Slave Address	0A
Function	04
Data	04 02 2B 00 64
CRC	F2

Table 11. Query Structure of Function16

Field Name	RTU
Header	None
Slave Address	11
Function	16
Starting Address Hi	00
Starting Address Lo	01
Quantity of Register Hi	00
Quantity of Register Lo	02
Byte Count	04
Data Hi	00
Data Lo	02
Data Hi	00
Data Lo	04
Error Check Lo	C6
Error Check Hi	B5

Table 12. Response Frame Structure of Function 16

Field Name	RTU
Header	None
Slave Address	11
Function	16
Starting Address Hi	00
Starting Address Lo	01
Quantity of Register Hi	00
Quantity of Register Lo	02
Error Check Lo	12
Error Check Hi	98

request data field. Here data packets are two bytes per register. Table 11 shows the frame structure of Query. Table 12 shows the frame structure of Response, it returns the slave address, function code, starting address and quantity of registers written.

3.2 Modbus Exception

Modbus exception codes are error codes that are returned by slave devices when it detects any problem with a command it received. It can be generated for several reasons, bad data values, invalid Modbus function codes, invalid register and much more. These errors result in an exception response from either the master computer software (Modbus Communications Handler) or the PC Slave, depending on the typeof error. The standardized Exception codes are listed in Table 13

Here we also focused on counting the type of error occur and in which situation. This helped us to understand our system much more deeply.

4. IMPLEMENTATION

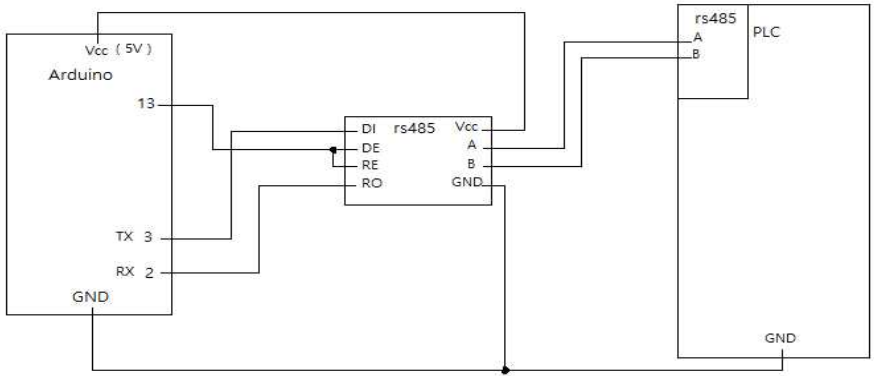
This proposed system was successfully implemented and tested with Arduino, RS485 and PLC by Modbus RTU protocol. Here we have discussed the implementation briefly with the result table. The system must confirm that devices on the same bus must have same communication parameters.

4.1 Wiring of the proposed system

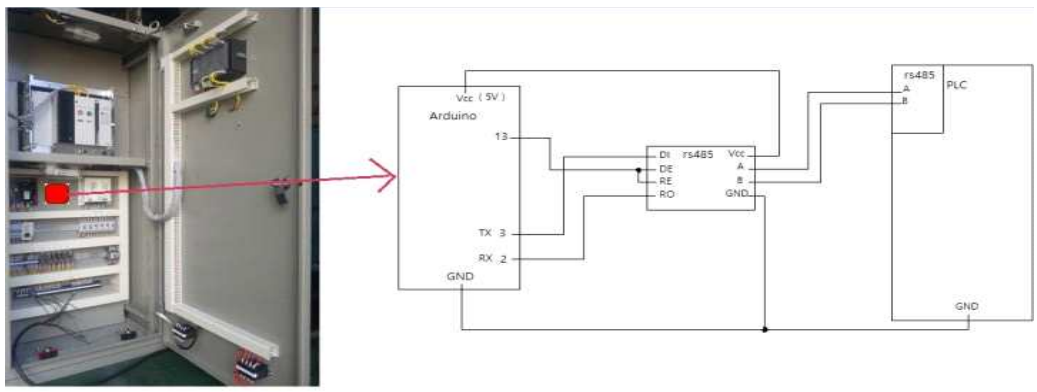
Fig. 1(a) gives an overall idea of the connection between Arduino and PLC with the mediator RS-485. Here TX and RX port of Arduino is con-

Table13. Response Frame Structure of Function 16

Code	Name	Description
01	Illegal function	The function value is not supported by the slave device
02	Illegal data Address	The Register Address value is not a valid register in the slave device.
03	Illegal Data Value	The Data value is not supported by the slave device register that is being written to.
04	Slave Device Failure	An error occurred in the slave device while attempting to perform a requested Modbus function query.
05	Acknowledge	The Slave device may return an Acknowledge response after receiving a function query that typically takes a long time to execute.
06	Slave Device Busy	The Slave device is busy processing a function or task.



(a)



(b)

(c)

Fig. 1. (a) System Wiring (b) Setup Location (c) PLC Setup Box.

nected to DI and RO of RS485 respectively. Whereas RE and DE port of RS485 are combined and connected to port13 of Arduino. Port 13 is used as a button.

Fig. 1(a) depicts the full system wiring. (b) depicts the location where we setup our system to test. (c) is the PLC setup Box where *39 and 40 port is connected to RS485(A, B port shown in Fig. 1(a))

Table 14. From Master to Slave

Address	Function Code	Data	Data	Data	Data	CRC Low	CRC HI
0×01	0×04	0×00	0×00	0×00	0×01	0×18	0×F0
0×01	0×04	0×00	0×01	0×00	0×01	0×19	0×60
0×01	0×04	0×00	0×02	0×00	0×01	0×19	0×90
0×01	0×04	0×00	0×03	0×00	0×01	0×19	0×00
0×01	0×04	0×00	0×04	0×00	0×01	0×1A	0×30

Table 15. Slave receives data

Address	Function Code	Data	Data	Data	CRC Lo	CRC HI
0×01	0×04	0×02	0×08	0×00	0×18	0×F0
0×01	0×04	0×02	0×04	0×00	0×19	0×60
0×01	0×04	0×02	0×02	0×00	0×19	0×90
0×01	0×04	0×02	0×01	0×00	0×19	0×00
0×01	0×04	0×02	0×00	0×80	0×1A	0×30

4.2 Results

To test our system we send data from Arduino to PLC using function code 04 where slave address is 0x01. Here starting data address from 0×00 to 0x04 represents the different data request from PLC.

In Table 14, first two data column states the starting address of PLC and next two data column states the amount of data is being sent. In the fourth row of this table, Data from 0×00-0×04 are the register start address from the data need to be read.

In the same way PLC reply to Arduino with data which was requested. We should note that the CRC should be same during each communication otherwise there will be an error in communication. Table 15 depicts the reply to Arduino request.

In Table 15, first data column lists the amount of data receives and the next two data column are received data. Third row Data represents the number of data is received. Fourth and fifth row represents the data of defined address.

5. CONCLUSION

We proposed and studied RS485 and MODBUS protocol for making communication between PLC and Arduino with less cost and time. We believe that this system will reduce the cost of development for various companies due to reusability and easy to install. This proposed system can be able to setup a communication network with a device having PLC installed in it or PLC supportable devices. Moreover, Arduino gives a wide range of option to connect other third party devices either through a wireless module or physically connected to it. Here we have mainly focused on error counting and handling. Study on error handling helped us to rectify the situation of error and on what type of data error has occurred. Due cost effective and complexity reduction this system will be helpful in the field of automatic devices.

In future, we are going to use this system as a mediator between two or more devices which can communicate each other effectively. This system will provide easy connectivity module for various devices.

REFERENCE

[1] Y. Jeong, M.I. Ansari, W. Shin, B. Kang, J. Shim, “Smart Disel Generator Control System,“ Journal of Korea Multimedia Society, Vol. 20, No. 2, pp. 271-278, 2017.

[2] B.B. Shibarinath and N. Gaur, "MODBUS Communication in Microcontroller Based Elevator Controller," *International Conference on Control, Automation, Robotics and Embedded Systems*, pp. 16-18, 2013.

[3] Comité Europeo de Normalización, *EN 61131-3: Programmable Controllers. Programming languages (IEC 611131-3:2013)*, International Electrotechnical Comission, Brussels, France, 2013.

[4] E. Tisserant, L. Bessard, and M. Sousa, "An Open Source IEC 61131-3 Integrated Development Environment," *Transactions on Industrial Informatics*, Vol. 1, pp. 183-187. 2007.

[5] M. Banzi, *Getting Started with Arduino,* O'Relly Media, Sebastopol, Calif., 2011.

[6] M. Margolis, *Arduino Cookbook,* O'Relly Media, Sebastopol, Calif., 2011.

[7] J. Ko, J. Shim, "A Comparison of the Construction for IoT System in Smart Clothing," *Journal of Multimedia Information System*, v.2, no.4, pp. 327-332, 2015.

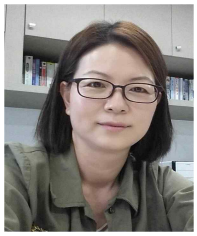
[8] G. Frey and L. Litz, "Formal Methods in PLC Programming," *Systems, Man, and Cybernetics, IEEE International Conference*, Vol. 4, No. 6, pp. 2431-2436, 2000.

[9] PLCopen Technical Committee 6, "XML Formats for IEC 61131-3, Ver 1.0", http://www.plcopen.org/pages/tc6_xml/downloads/tc6_xml_v201_technical_doc.pdf (accessed Nov., 21, 2016).

[10] MODBUS over Serial Line Specification and Implementation Guide V1.02, http://www.modbus. org/docs/Modbus_over_serial_line_V1_02.pdf (accessed Nov., 21, 2016).

[11] X. Jin, H. Yuan, Q. Jiang, and Y. Wang, "Design of Silicon-based Transient Voltage Suppressor to Meet IEC System-level ESD Specification for RS485 Transceiver," *Proceeding of International Conference on Solid-State and Integrated Circuit Technology*, pp. 1-3, 2014.

[12] Z. Wu, S. Liu, M. Du, Q. Li, C. Han, and J. Wu, "Design of Vibrating Wire Sensor Singnal Acquisition Board Based on STM32," *Proceeding of World Congress on Intelligent Control and Automation,* pp. 1846-1849, 2016.



Yunju Jeong

received BS in Computational Statistics from Andong National University in 1991 and Master of Computer Engineering from Department of Computer Engineering, Andong National University in 2000. Currently, she studies for Ph.D. at Department of Computer Engineering of Kyungpook National University and also an Education Instructor at Department of Computer Engineering, Department of Electronic Engineering, Andong National University, Korea. Her research interests include computer vision, Deep Learning, and VR.

Md Israfil Ansari

received the BCA degree in Computer Application from Punjab Technical University, Jalandhar, India in 2011 and the MCA degree in Computer Application from Visvesvaraya Technological University, Bangalore, India in 2015. His previous position was a Software Developer as an Intern at Xcelvations Consultancy Pvt Ltd, Hyderabad, India. Currently studies for Ph.D. at Department of Computer Engineering, Andong National University, Korea. He has interests in the areas of computer vision, pattern recognition, Deep Learning.

WooHyeon Shin

currently is a B.S student at Department of Computer Engineering, Andong National University. His research interests include computer vision, programming language, and IoT.

Bonggu Kang

received B.S.degree in 2015 and M.Eng degree on Feb 2017 from Department of Computer Engineering, Andong National University, Korea. His research interests include Embedded vision and Android applications.

JinSeop Lim

received B.S.degree in 2015 and M.Eng degree on Feb 2017 from Department of Computer Engineering, Andong National University, Korea. His research interests include programming, database, windows programming.

HyeonSik Moon

currently is a B.S student at Department of Computer Engineering, Andong National University. His research interests include IoT, Ubiquitous environment construction and Electronic circuit

Jaechang Shim

received the B. S. degree in 1987 and Ph.D. degree from Kyungpook National University, Korea, in 1997. From 1997 to 1999, he worked as a researcher in IBM T. J. Watson Research Center. From 2005 to 2006, he was a visiting Fellow Professor at Princeton University. Since 1994, he has been a Professor of the Andong National University, Korea. He received the best Industrial-Related Paper Award form International Association for pattern Recognition in 1998. His research interests include computer vision, image processing, and pattern recognition.